

(19)



JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: 09330219 A

(43) Date of publication of application: 22.12.97

(51) Int. Cl

G06F 9/30
G06F 9/45

(21) Application number: 08151260

(71) Applicant: MATSUSHITA ELECTRIC IND CO LTD

(22) Date of filing: 12.06.96

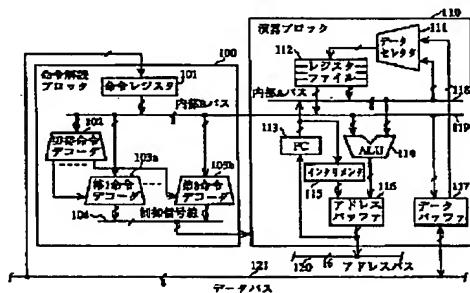
(72) Inventor: TAKAYAMA SHUICHI
HIGAKI NOBUO
TOMINAGA NOBUTERU
**(54) INSTRUCTION DECODER SWITCHING
PROCESSOR AND TRANSLATOR**

(57) Abstract:

PROBLEM TO BE SOLVED: To provide a processor with which many kinds of machine word instructions can be defined without increasing an instruction word length.

SOLUTION: Concerning the processor composed of an instruction decode block 100 and an arithmetic block 110, the instruction decode block 100 is further composed of an instruction register 101 for holding fetched instructions, eight different kinds of instruction decoders 103a-103h for decoding the fetched instructions and controlling the arithmetic block 110, and switching instruction decoder 102 for validating only one of these eight instruction decoders 103a-103h according to a fetched dedicated switching instruction.

COPYRIGHT: (C)1997,JPO



(19)日本国特許庁 (JP)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開平9-330219

(43)公開日 平成9年(1997)12月22日

(51) Int.Cl.⁶
G 06 F 9/30
9/45

識別記号 310
序内整理番号

F I
G 06 F 9/30
9/44

技術表示箇所
310 C
322 F

審査請求 未請求 請求項の数14 O.L (全 15 頁)

(21)出願番号 特願平8-151260
(22)出願日 平成8年(1996)6月12日

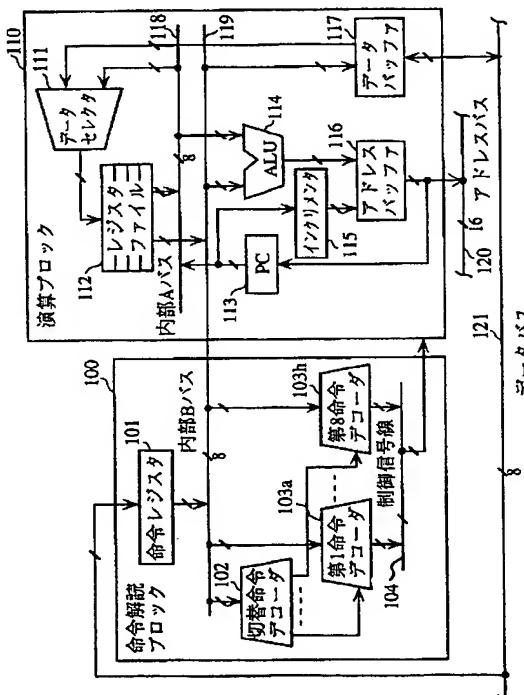
(71)出願人 000005821
松下電器産業株式会社
大阪府門真市大字門真1006番地
(72)発明者 高山 秀一
大阪府門真市大字門真1006番地 松下電器
産業株式会社内
(72)発明者 檜垣 信生
大阪府門真市大字門真1006番地 松下電器
産業株式会社内
(72)発明者 富永 宣輝
大阪府門真市大字門真1006番地 松下電器
産業株式会社内
(74)代理人 弁理士 中島 司朗

(54)【発明の名称】 命令解読器切替型プロセッサ及び翻訳装置

(57)【要約】

【課題】 命令語長を増加させずに多くの種類の機械語命令を定義することができるプロセッサを提供する。

【解決手段】 命令解読ブロック100と演算ブロック110とから構成されるプロセッサであって、命令解読ブロック100はさらに、フェッチした命令を保持する命令レジスタ101と、フェッチされた命令を解読し演算ブロック110を制御する8個の異なる種類の命令デコーダ103a～103hと、フェッチされた専用の切替命令によってそれら8個の命令デコーダ103a～103hの中から1個だけを有効とさせる切替命令デコーダ102とからなる。



1

【特許請求の範囲】

【請求項 1】 命令メモリから命令を読み出して解読する命令解読手段と、その命令解読手段から出力される解読結果に従って演算処理を実行する実行手段とを備えるプロセッサであって、
前記命令解読手段は、

前記命令メモリから命令を読み出すフェッチ手段と、
前記フェッチ手段により読み出された命令を解読し、少なくとも 1 個の同一命令に対して異なる解読結果を出力する n (2 以上の整数) 個の命令解読器と、
前記 n 個の命令解読器から選択された一つの命令解読器の解読結果だけが前記実行手段に出力されるよう前記 n 個の命令解読器を制御する制御手段とからなることを特徴とする命令解読器切替型プロセッサ。

【請求項 2】 前記制御手段は、
前記フェッチ手段により読み出された命令が所定の命令解読器切替命令であるかどうかを判断する判断部と、
前記判断部により所定の命令解読器切替命令であると判断された場合には、その命令解読器切替命令に基づいて前記選択を行う選択部とからなることを特徴とする請求項 1 記載の命令解読器切替型プロセッサ。

【請求項 3】 前記選択部は、
前記判断部により所定の命令解読器切替命令であると判断された場合には、その命令解読器切替命令から一の命令解読器を特定する情報を抽出する抽出部と、
前記抽出部により抽出された情報を記憶する記憶部と、
前記 n 個の命令解読器のうち前記記憶部に記憶された情報に対応する命令解読器の解読結果だけが前記実行手段に出力されるよう前記 n 個の命令解読器を制御する出力制御部とからなることを特徴とする請求項 2 記載の命令解読器切替型プロセッサ。

【請求項 4】 前記出力制御部は、
前記 n 個の命令解読器の解読結果が前記実行手段に出力されることを命令解読器ごとに許可又は禁止するゲート部と、
前記記憶部に記憶された情報に基づいて前記ゲート部を制御するゲート制御部とからなることを特徴とする請求項 3 記載の命令解読器切替型プロセッサ。

【請求項 5】 前記 n は 2 であることを特徴とする請求項 4 記載の命令解読器切替型プロセッサ。

【請求項 6】 プログラムを翻訳することにより、n (2 以上の整数) 個の異なる種類の命令解読器を備えるプロセッサを対象とする機械語命令列を生成する翻訳装置であって、
最適な機械語命令列を生成するためには前記 n 個の命令解読器の中のいずれを対象とすべきかを決定する決定手段と、
前記決定手段により決定された命令解読器を対象として前記プログラムを翻訳し機械語命令列を生成する生成手段とを備えることを特徴とする翻訳装置。

2

【請求項 7】 前記決定手段は、生成された機械語命令列のコードサイズが最小となる場合を最適とすることを特徴とする請求項 6 記載の翻訳装置。

【請求項 8】 前記決定手段は、生成された機械語命令列を前記プロセッサが実行するのに要する時間が最小となる場合を最適とすることを特徴とする請求項 6 記載の翻訳装置。

【請求項 9】 前記翻訳装置はさらに、前記プログラムを所定の基本単位に分割する分割手段を備え、

10 前記決定手段及び前記生成手段は、前記分割手段により分割された基本単位ごとに前記決定及び前記生成を行うことを特徴とする請求項 7 又は 8 記載の翻訳装置。

【請求項 10】 プログラムを翻訳することにより、n (2 以上の整数) 個の異なる種類の命令解読器を備えるプロセッサを対象とする機械語命令列を生成する翻訳装置であって、

前記プログラムから前記 n 個の命令解読器のそれぞれを対象とする機械語命令列とその最適化に必要な情報を生成する暫定翻訳手段と、

20 前記暫定翻訳手段により生成された前記情報に基づいて前記 n 個の機械語命令列の中から最適な 1 個を決定しその機械語命令列をこの翻訳装置が生成する機械語命令列とする決定手段とを備えることを特徴とする翻訳装置。

【請求項 11】 前記暫定翻訳手段は、生成した n 個の機械語命令列のそれぞれに対応するコードサイズを前記情報として生成し、

前記決定手段は、生成されたコードサイズが最小となる機械語命令列を最適なものとして決定することを特徴とする請求項 10 記載の翻訳装置。

30 【請求項 12】 前記暫定翻訳手段は、生成した n 個の機械語命令列のそれぞれを前記プロセッサが実行するのに要する時間を前記情報として生成し、
前記決定手段は、生成された実行時間が最小となる機械語命令列を最適なものとして決定することを特徴とする請求項 10 記載の翻訳装置。

【請求項 13】 前記翻訳装置はさらに、前記プログラムを所定の基本単位に分割する分割手段を備え、
前記暫定翻訳手段及び前記決定手段は、前記分割手段により分割された基本単位ごとに前記生成及び前記決定を行うことを特徴とする請求項 11 又は 12 記載の翻訳装置。

【請求項 14】 前記 n は 2 であることを特徴とする請求項 9 又は 13 記載の翻訳装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 本発明は、複数の命令解読器を備えたプロセッサ及びそのコンパイラに関し、特に機械語命令列のコードサイズの縮小化を支援するプロセッサ及びコンパイラに関する。

50 【0002】

【従来の技術】近年、マイクロプロセッサ（以下、単に「プロセッサ」という。）を用いた製品は、ますます高機能化が求められ、それに伴ってオブジェクトプログラム（以下、単に「オブジェクト」という。）は大規模化してきている。ところが、プロセッサ組み込み型の製品においては、オブジェクトをROMに格納して実装する必要があるため、そのコードサイズの増大は実装すべきROMの容量や個数の増大をもたらし、低価格な製品の開発を阻害する。従って、このような製品の開発においては、可能な限りコードサイズを縮小しておくことが望まれる。また、処理速度の高速化の点においても、より少ないステップ数のオブジェクトが望まれる。

【0003】コードサイズの縮小化を支援する従来技術として、命令セットの定義において、使用頻度の高い機械語命令（以下、単に「命令」という。）を語長の短い命令に割り当てておく方法が挙げられる。具体的には、例えば、1バイト長命令と2バイト長命令の2種類の語長の命令からなる命令セットを備えるプロセッサであれば、使用頻度が高いと考えられる命令を1バイト長命令に割り当てておく方法である。

【0004】図17は、従来のプロセッサが備える命令のフォーマットの一例を示す図である。図17(a)は1バイト長命令、図17(b)は2バイト長命令のフォーマットを示す。なお、ここでは、命令の語長とは、命令を構成するオペコードとオペランドのうちのオペコードだけの長さをいうものとする。また、オペコードは、プロセッサにフェッチされた後に命令デコーダによって解読される部分であり、その命令の種類（演算処理の種類）を特定するコードである。

【0005】このように、命令の種類はそのオペコードの長さによって決定されるので、例えば、図17(a)に示された1バイト長命令としては最大で256種類の命令が定義される。従って、使用頻度が高いと考えらる256個の命令を1バイト長命令に割り当ててプロセッサの命令デコーダを設計しておくことによって、同一処理内容を指示するオブジェクトであっても、そのような考慮をしていない場合に比べ、コードサイズを縮小することができる。

【0006】

【発明が解決しようとする課題】しかしながら、プロセッサの高機能化が求められる今日においては、上記種類の命令だけでは十分とは言えない。特に、近年のマルチメディア技術の発展に伴い、画像データ処理関連（データ圧縮・伸長、符号・復号、積和演算等）の命令の必要性が高まっているが、従来のプロセッサでは、例えば、画像処理関連の新たな命令を従来の基本命令セットに追加するのは困難である。

【0007】即ち、上記のような従来のプロセッサであれば、256種類を超える1バイト長命令を備えることはもはや不可能である。従って、従来の256種類の1

バイト長命令を維持したままで、新たな種類の命令を追加するためには、2バイト長命令又は2バイトを超える拡張命令を定義する必要があるが、これでは、コードサイズの縮小化が阻害される。

【0008】そこで、本発明は、かかる問題点に鑑みてなされたものであり、一定語長の命令であっても、より多くの種類の命令を定義することができるプロセッサ及びそのための翻訳装置を提供することを目的とする。具体的には、1バイト長命令であれば、実質的に256種類を超える命令を備えるプロセッサ、また、2バイト長命令であれば、実質的に65536種類を超える命令を備えるプロセッサ及びそのための翻訳装置を提供することを目的とする。

【0009】

【課題を解決するための手段】上記目的を達成するため、本発明に係るプロセッサは、命令メモリから命令を読み出して解読する命令解読手段と、その命令解読手段から出力される解読結果に従って演算処理を実行する実行手段とを備えるプロセッサであって、前記命令解読手段は、前記命令メモリから命令を読み出すフェッチ手段と、前記フェッチ手段により読み出された命令を解読し、少なくとも1個の同一命令に対して異なる解読結果を出力するn(2以上の整数)個の命令解読器と、前記n個の命令解読器から選択された一つの命令解読器の解読結果だけが前記実行手段に出力されるよう前記n個の命令解読器を制御する制御手段とからなることを特徴とする。

【0010】これによって、例えば、1バイト長命令であれば実質的に256種類を超える命令を備えるプロセッサ、また、2バイト長命令であれば実質的に65536種類を超える命令を備えるプロセッサが実現されるので、従来の单一デコーダ型プロセッサに比べ、同一処理を行わせる機械語プログラムであってもそのコードサイズ及び実行時間は短縮される。

【0011】また、本発明に係る翻訳装置は、プログラムを翻訳することにより、n(2以上の整数)個の異なる種類の命令解読器を備えるプロセッサを対象とする機械語命令列を生成する翻訳装置であって、最適な機械語命令列を生成するためには前記n個の命令解読器の中のいずれを対象とすべきかを決定する決定手段と、前記決定手段により決定された命令解読器を対象として前記プログラムを翻訳し機械語命令列を生成する生成手段とを備えることを特徴とする。

【0012】さらに、本発明に係る翻訳装置は、プログラムを翻訳することにより、n(2以上の整数)個の異なる種類の命令解読器を備えるプロセッサを対象とする機械語命令列を生成する翻訳装置であって、前記プログラムから前記n個の命令解読器のそれぞれを対象とする機械語命令列とその最適化に必要な情報を生成する暫定翻訳手段と、前記暫定翻訳手段により生成された前記

情報に基づいて前記n個の機械語命令列の中から最適な1個を決定しその機械語命令列をこの翻訳装置が生成する機械語命令列とする決定手段とを備えるとともにできる。

【0013】これによって、複数の命令デコーダ、即ち、複数の命令セットを備えるプロセッサを対象とする翻訳装置が実現され、高級言語等による上記プロセッサの利用が可能となる。

【0014】

【発明の実施の形態】以下、本発明に係る命令解読器切替型プロセッサ及びその翻訳装置について、図面を参照しながら具体的に説明する。

(実施の形態1) 実施の形態1では、本発明に係る命令解読器切替型プロセッサについて説明する。

【0015】図1は、本発明の実施の形態1における命令解読器切替型プロセッサの構成を示すブロック図である。本プロセッサは、大きく分けて、命令解読ブロック100と演算ブロック110とからなる。

【命令解読ブロックの構成】命令解読ブロック100は、データバス121を介してフェッチした命令を解読する部分であり、命令レジスタ101、切替命令デコーダ102及び8個の命令デコーダ103a～103hから構成される。なお、本図には、8個の命令デコーダ103a～103hのうち、2個の命令デコーダ(第1命令デコーダ103a及び第8命令デコーダ103h)だけが図示され、他の6個の命令デコーダ(第2命令デコーダ103b～第7命令デコーダ103g)の図示は省略されている。また、各構成要素101～103hは、図示されていないクロック信号や制御信号の供給を受けて同期して動作する。

【0016】命令レジスタ101は、1バイト長のレジスタであり、図示されていない外部メモリからデータバス121を介して順次にフェッチした1バイト分の命令を保持する。ここに保持された命令は、オペコードであれば切替命令デコーダ102と8個の命令デコーダ103a～103hの合計9個の命令デコーダに同時に送られ、一方、オペランドであれば演算ブロック110に転送される。なお、説明の容易のため、本プロセッサは、1種類の命令フォーマット、具体的には、図17(a)に示された1バイト長命令のみを備えるものとする。

【0017】切替命令デコーダ102は、8個の命令デコーダ103a～103hのうちの1個を有効とさせるための制御回路であり、図2に示されるように、さらに、切替命令専用解読器102a及びラッチ102bから構成される。切替命令専用解読器102aは、比較器とデコーダ等からなり、命令レジスタ101から送られてきた命令が切替命令であるかどうかを判断し、切替命令である場合には、その命令の一部をデコードすることにより、8個の命令デコーダ103a～103hのうちの1個のみを有効化させる8ビットのデータパターンを

ラッチ102bに保持させる。

【0018】ここで、切替命令とは、図3に示される8種類の命令のいずれかをいい、切替命令であることを示す5ビットと8個の命令デコーダ103a～103hのうちの1個を特定する3ビットからなる。8個の命令デコーダ103a～103hは、命令レジスタ101から送られてきた1バイト長命令を同時に解読し、それらの解読結果は、切替命令デコーダ102のラッチ102bからの信号により、いずれか1つのみが選択されて制御信号線104に出力される。

【0019】各命令デコーダ103a～103hは、いずれも、図4に示されるように、1バイト幅を基本とするプリデコーダ401、シーケンサ402、マイクロROM403及び出力バッファ404から構成され、形式的な構成はそれぞれ同一である。プリデコーダ401は、入力された1バイト長のオペコードを、対応するマイクロコードが格納されたマイクロROM403のエントリアドレ스に変換するものである。シーケンサ402は、プリデコーダ401が示すエントリアドレ스から始まるマイクロコードをマイクロROM403から出力させるためのアドレス発生用カウンタである。マイクロROM403は、各命令に対応するマイクロコードを記憶している。出力バッファ404は、ゲート回路であり、切替命令デコーダ102のラッチ102bからのゲート信号による制御の下に、マイクロROM403から出力されたマイクロコードを制御信号線104に出力したり、禁止したり(出力をハイインピーダンスにする)する。

【0020】ここで、8個の命令デコーダ103a～103h相互の関係は、以下の通りである。8個の命令デコーダ103a～103hは、いずれも256種類の1バイト長命令が定義された命令セットを有しているが、それら命令セットは、完全には一致しない。具体的には、各命令デコーダ103a～103hが備えるマイクロROM403は、256種類の命令のうちの一部(ここでは、32種類)の命令に対するマイクロコードが異なり、他の多く(ここでは、224個)の命令に対するマイクロコードが一致する。そのために、同一のオペコードを有する命令を解読した場合であっても、各命令デコーダ103a～103hは、異なる解読結果を出力することがある。即ち、同一のオペコードを有する命令が本プロセッサにフェッチされた場合であっても、その時点で有効とされている命令デコーダの種類によっては制御信号線104に出力される信号が異なり、演算ブロック110での動作が異なることがある。

【0021】ここで、上記32種類の命令、即ち、オペコードが同一であるが解読される命令デコーダに依存してプロセッサの動作が異なるようなものを「デコーダ依存型命令」、一方、残る224種類の命令、即ち、解読される命令デコーダに依存することなくプロセッサの動作が同一となるものを「デコーダ非依存型命令」と呼

ぶ。なお、デコーダ依存型命令の数をデコーダ非依存型命令よりも少なくしているのは、デコーダ依存型命令の数を多くした場合に生ずるマイナス効果、即ち、有効とする命令デコーダを頻繁に切り替えなければならないことから生ずるオーバーヘッド（処理速度の低下）を回避するためである。

【0022】図5は、本プロセッサの命令セット、即ち、上記8個の命令デコーダ103a～103hを一体とした場合の命令セットの一部を示すテーブルである。なお、本図で用いられている、“opn”はオペランド、“label”は分岐先を示すラベルを意味する。図5に示された命令の中では、“sp1”だけがデコーダ依存型命令であり、解読される命令デコーダの種類によってプロセッサの動作が異なる。なお、本図には、これら異なる8種類の動作のうち2種類だけが示されている。ここで、ゼロラン長とは、ビット並びにおいて初めて「1」が出現する場合のその位置をいい、例えば、ビット並び「00110100」であれば最下位ビット（最右端）からのゼロラン長は3である。この命令は、画像認識処理等において必要とされる命令の一つである。

【演算ブロックの構成】演算ブロック110は、命令解読ブロック100での解読結果に従って演算やデータ転送等の処理を実行する部分であり、データセレクタ111、レジスタファイル112、プログラムカウンタ113、ALU114、インクリメンタ115、アドレスバッファ116及びデータバッファ117から構成される。

【0023】この演算ブロック110は、通常のプロセッサが備えるものと異ならない。各構成要素111～117は、命令解読ブロック100からの制御信号線104に接続され、その信号によって演算やデータ転送等の処理を行う。データセレクタ111は、内部Aバス118か又はデータバッファ117からのデータを選択してレジスタファイル112に送る。レジスタファイル112には、スタックポインタや16個の汎用レジスタ等が含まれる。プログラムカウンタ113は、命令解読ブロック100にフェッチされた命令が置かれていた外部メモリ上のアドレス等を保持する。プログラムカウンタ113に保持されたアドレスは、インクリメンタ115によって1だけ加算され、次にフェッチすべき命令のアドレスとしてアドレスバッファ116を介してアドレスバス120に出力されたり、プログラムカウンタ113に送られたりする。ALU114は、内部Aバス118及び内部Bバス119のそれぞれから送られてくるデータの算術演算を行う。内部Bバス119は、主に、命令レジスタ101に保持されたオペランドを演算ブロック110に転送するのに用いられる。

【命令解読器切替型プロセッサの動作】以上のように構成された本プロセッサの動作について、具体的な命令を実行する場合を例にとって説明する。

【0024】図6は、実行の対象となるオブジェクトのリストである。このリストで用いられている個々の命令の意味は、図5に示されるテーブルの通りである。また、オペランドに用いられているアドレッシングモードの意味は、以下の通りである。

- a : 変数aの値
- &b : 変数bが格納されている外部メモリ上のアドレス
- (c) : 外部メモリ上のアドレスcに格納されている値

図6に示されたオブジェクトは、大きく分けて、ラベル_m、_f及び_gで始まる3つのブロックからなる。

【0025】ラベル_mから始まるブロック（ステップS600～S604）は、c言語でいうメイン関数に相当し、引き数t1を設定した後に関数fを呼び出し、続いて、再び引き数t1を設定した後に関数gを呼び出すものである。なお、関数fでの処理と関数gでの処理は特に関連するものではない。ラベル_fから始まるブロック（ステップS605～S607）は、関数fに相当し、データブロックのコピーを行うものである。

【0026】ラベル_gから始まるブロック（ステップS608～S416）は、関数gに相当し、データブロックの各要素についてのゼロラン長を比較するものである。このようなオブジェクトを実行した場合の本プロセッサの具体的な動作は、以下の通りである。なお、このオブジェクトの実行に先立ち、切替命令デコーダ102は初期化され、第1命令デコーダ103aが有効とされているものとする。

【ステップS600、S601】プロセッサは、第1命令デコーダ103aを用いて、関数mの先頭から実行を開始し関数fを呼び出しが、これらのステップにおいては、結果的には1個の命令デコーダのみからなる従来のプロセッサ（以下、「單一デコーダ型プロセッサ」と呼ぶ。）の場合と同一の動作をする。

【ステップS605】命令“ex8”は、切替命令デコーダ102によって解読され、その結果、ラッチ102bに保持されるパターンは、第8命令デコーダ103hを有効とするパターンに更新される。これによって、次の命令（ステップS606）からは、第8命令デコーダ103hによって解読されることになる。

【0027】なお、切替命令は、他の命令と同様に、切替命令デコーダ102に解読されると同時に、各命令デコーダ103a～103hによっても解読されるが、いずれの命令デコーダも、切替命令に対してはノーオペレーション動作となるよう設計されている。

【ステップS606】命令“sp1”は、第8命令デコーダ103hによって解読されるので、その結果、図5に示されるように、アドレスbから始まる外部メモリ上の10個のデータがアドレスt1から始まる領域にコピーされる。

【ステップ S 607、S 602、S 603】次に、プロセッサは、関数 f での処理を終えて関数 m に戻った後に関数 g を呼び出すが、これらのステップにおいても、結果的には従来の単一デコーダ型プロセッサの場合と同一の動作をする。

【ステップ S 608】命令 "ex1" は、切替命令デコーダ 102 によって解読され、その結果、ラッチ 102b に保持されるパターンは、第 1 命令デコーダ 103a を有効とするパターンに更新される。これによって、次の命令 (ステップ S 609) からは、再び第 1 命令デコーダ 103a によって解読されることになる。

【ステップ S 609～S 616、S 604】ステップ S 610においては、再び命令 "sp1" が実行されるが、ステップ S 606 の場合と相違し、ここでは第 1 命令デコーダ 103a によって解読されるので、図 5 に示されるように、アドレス a から i 番目に位置する外部メモリ上のデータのゼロラン長が算出され変数 j に書き込まれる。

【0028】ステップ S 611における動作もステップ S 610の場合と同様であり、アドレス b から i 番目に位置する外部メモリ上のデータのゼロラン長が算出され変数 k に書き込まれる。このように、第 8 命令デコーダ 103h が有効とされていたステップ S 606 と第 1 命令デコーダ 103a が有効とされていたステップ S 610 及び S 611 における動作を比較して判るように、本プロセッサは、同一オペコードの命令 "sp1" に対して、2種類の異なる動作を行った。

【0029】なお、ステップ S 609～S 615においては、アドレス a から始まる 10 個のデータとアドレス b から始まる 10 個のデータのそれぞれについてゼロラン長を比較し、不一致が発見された場合には、図示されていない関数 error を呼び出す処理が行われる。以上のように、本プロセッサは、1 バイト長のオペコードで定義される命令セット、即ち、形式的に 256 種類の命令を備えるが、それら命令の一部はデコーダ依存型命令であることから、実質的に 256 種類を超える命令を備えることに等しい。

【0030】具体的には、本プロセッサは、224 個のデコーダ非依存型命令と 32 個のデコーダ依存型命令を有するので、1 個のデコーダ依存型命令が実質的に 8 個の命令に相当することから、合計 480 (= 224 + 32 × 8) 個の命令を備えることに等しい。即ち、本プロセッサの如く複数の命令解読デコーダを備えることによって、1 バイト長命令のフォーマットを維持したままで、従来の 256 種類の命令からなる命令セットに加えて、さらに、224 (= 480 - 256) 種類の新たな命令を定義することが可能となる。

【0031】また、本プロセッサを構成する各命令デコーダは、相互に形式的な構成が同一であり、しかも、従来のプロセッサの命令デコーダと同様に、8 ビット幅の

命令を解読するものである。従って、従来のプロセッサが有する解読器をそのまま流用する形で、大幅に設計変更することなく、本プロセッサを容易に構築することができる。

【0032】さらに、本プロセッサにおいては、切替命令デコーダ 102 と他の 8 個の命令デコーダ 103a～103h とは、その処理内容が異なる独立した回路であるため、パイプライン構成とすることも容易である。なお、本実施形態においては、プロセッサは、8 個の命令デコーダを備え、1 種類の 1 バイト長命令だけを実行したが、本発明は、これら数値に限定されるものではない。例えば、プロセッサは、2 個の命令デコーダを備え、2 バイト長命令を実行し、各命令デコーダは、それぞれ 32 種類のデコーダ依存型命令と 65504 種類のデコーダ非依存型命令を備えるとすることもできる。これによって、2 バイト長命令のフォーマットを維持したままで、従来の 65536 種類の命令からなる命令セットに加えて、さらに、 $32 (= 32 \times 2 + 65504 - 65536)$ 種類の新たな命令を定義することが可能となる。

【0033】また、本実施形態においては、8 個の命令デコーダのそれぞれのマイクロ ROM は、共通する部分 (256 個のうちの 224 個の命令に相当する部分) を冗長に有しているが、このような構成に限られるものではない。例えば、8 個の命令デコーダは、相互に異なる部分 (256 個のうちの 32 個の命令に相当する部分) だけをマイクロ ROM に有し、共通する部分は、これら 8 個の命令デコーダとは異なる別の 1 個の命令デコーダのマイクロ ROM に格納されるとすることもできる。即ち、デコーダ依存型命令だけが 8 個の命令デコーダのいずれかによって解読され、一方、デコーダ非依存型命令は共通の 1 個の命令デコーダによって解読されることとなる。これによって、命令デコーダの切替頻度は多少増えるものの、必要とされるハードウェア回路を削減することができる。

(実施の形態 2) 実施の形態 2 では、本発明に係る命令解読器切替型プロセッサの他の実現形態について説明する。

【0034】本実施形態におけるプロセッサは、実施形態 1 におけるプロセッサと同一の機能、即ち、同一の命令セットを備えるが、その実現方法が異なる。具体的には、本実施形態におけるプロセッサは、大きく分けて命令解読ブロックと演算ブロックの 2 個のブロックからなる点において、実施形態 1 のプロセッサと同じであるが、命令解読ブロックの内部構成が異なる。以下、実施形態 1 と異なる点を中心に説明する。

[命令解読ブロックの構成] 図 7 は、本発明の実施の形態 2 における命令解読器切替型プロセッサの命令解読ブロック 700 のみの構成を示すブロック図である。

【0035】命令解読ブロック 700 は、命令レジスタ

701、デマルチプレクサ 702 及び 8 個の命令デコーダ 703a～703h から構成され、命令レジスタ 701 から出力された 1 バイト長の命令は、デマルチプレクサ 702 によって 8 個の命令デコーダ 703a～703h のいずれか 1 個にのみ振り分けられる点で、実施形態 1 と異なる。

【0036】デマルチプレクサ 702 は、図 8 に示されるように、さらに、ラッチ 702a 及び切替器 702b からなる。ラッチ 702a は、現時点で有効とする命令デコーダを特定する情報、即ち、8 個の命令デコーダ 703a～703h のいずれか 1 個から出力された切替信号を保持し、切替器 702b を制御する。切替器 702b は、ラッチ 702a からの制御の下に、命令レジスタ 701 からの 1 バイト長命令を 8 個の命令デコーダ 703a～703h のいずれか 1 個に通過させ、他の 7 個の命令デコーダに対してはデータ "f f (hex)" を出力する。

【0037】8 個の命令デコーダ 703a～703h は、いずれも、図 9 に示されるように、さらに、1 バイト幅を基本とするプリデコーダ 901、シーケンサ 902、マイクロ ROM 903 及び出力バッファ 904 から構成され、形式的な構成はそれぞれ同一である。これら命令デコーダと実施形態 1 のものとの主な相違点は以下の 2 点である。

【0038】即ち、第 1 に、プリデコーダ 901 は、内臓する比較器により、データ "f f (hex)" が入力されたときには、出力バッファ 904 からの出力を禁止する。これは、自己の命令デコーダが選択されていないと判断した場合に相当する。一方、"f f (hex)" を除くデータが入力されたときには、プリデコーダ 901 は、実施形態 1 の場合と同様の動作を行う。これは、自己の命令デコーダが有効であると選択されている場合に相当する。

【0039】第 2 に、8 個の命令デコーダ 703a～703h が備える全てのマイクロ ROM 903 は、8 種類の切替命令に対応するマイクロコードを有している。以上のように構成された命令解読ブロック 700 により、実施形態 1 の場合と異なり、命令レジスタ 701 に保持された命令は、8 個の命令デコーダ全てによって解読されるのではなく、有効とされている 1 個の命令デコーダにのみ送られて解読されることになる。これによって、有効とされていない 7 個の命令デコーダでの無駄な解読動作が回避され、また、切替命令だけを解読する専用のデコーダが不要となる。

【0040】なお、実施形態 1 及び 2 においては、各命令デコーダ自体は、マイクロプログラム方式により実現されたが、この方式に代えて、RISC (Reduced Instruction Set Computer) 等に採用されているワイヤードロジック方式により実現されてもよい。即ち、本発明は、命令デコーダ自体の実現方法を特徴とするものでは

なく、第 1 に、複数の命令デコーダ、即ち、複数の命令セットを備えることを特徴とし、第 2 に、最後に解読・実行した切替命令の種類、即ち、その時点におけるプロセッサの状態に依存して、それら複数の命令セットの中から使用すべき 1 個の命令セットが決定されることを特徴とするものである。

(実施の形態 3) 次に、本発明に係る翻訳装置の一例、ここでは、上記実施形態 1 のプロセッサをターゲットとする翻訳装置について説明する。

【0041】図 10 は、本実施形態における翻訳装置の構成を示す機能ブロック図である。本装置は、汎用のコンピュータシステム及びその上で実行されるプログラムによって実現されるものであり、機能的には、ブロック分割部 1001、中間コード生成部 1002、コード生成部 1003、オブジェクト決定部 1004、パラメータ獲得部 1005 及び命令テーブル 1006 から構成される。なお、図 10 には、これら各構成要素 1001～1006 の処理対象となる入出力データ 1010～1022 も同時に示されている。

【0042】パラメータ獲得部 1005 は、操作者からの指示に従って、本装置が有する選択的な機能（オプション）を指定する情報を獲得し、その情報をブロック分割部 1001 及びオブジェクト決定部 1004 に通知する。オプションには分割パラメータと最適化パラメータの 2 種類がある。「分割パラメータ」は、使用される命令デコーダ（命令セット）の種類が切り替えられることがない最小のプログラム単位（以下、「ブロック」という。）の種類（例えば、「関数」、途中に分岐の入口出口を持たない「基本ブロック」等）を指定するものであり、一方、「最適化パラメータ」は、オブジェクトを最適化する際の基準（例えば、「コードサイズ」、「クロックサイクル数」等）を指定するものである。

【0043】ブロック分割部 1001 は、パラメータ獲得部 1005 から通知される分割パラメータに基づいて、c 言語等で書かれたソースプログラム 1010 を複数のブロック 1011～1013 に分割する。中間コード生成部 1002 は、分割された各ブロック 1011～1013 ごとに、字句・構文解析をした後に 3 オペランド表現による中間コードを生成することにより、各ブロックに対応する中間ファイル 1014～1016 を出力する。

【0044】命令テーブル 1006 は、図 11 に示される情報を予め保持しているものであり、中間コード生成部 1002 が使用し得る全ての種類の中間コード 1101 と、各中間コードに対応する 8 組の情報 1102 とかなる。この 8 組の情報 1102 は、8 個の命令デコーダ 103a～103h のそれぞれをターゲットとした場合における各中間コードに対応する命令、その命令のサイズ（バイト数）及びその命令の解読・実行に要するクロックサイクル数からなる。

【0045】コード生成部1003は、中間コード生成部1002によって生成された各中間ファイル1014～1016ごとに、命令テーブル1006を参照することにより、各命令デコーダ103a～103hに対応する8種類のオブジェクト1017a～1017h、1018a～1018h、1019a～1019hを生成する。このとき、コード生成部1003は、命令テーブル1006を参照することによって、各中間コードを命令に変換するだけでなく、各中間コードに対応する全てのサイズ及びクロックサイクル数の合計値をも算出して各オブジェクトに書き込んでおく。

【0046】オブジェクト決定部1004は、ブロックごとに、コード生成部1003によって生成された8種類のオブジェクトから最適な1種類を決定し、決定したそのオブジェクトをそのブロックの最終オブジェクトとして出力する。このとき、オブジェクト決定部1004は、パラメータ獲得部1005から通知される最適パラメータに基づき、合計サイズ又は合計クロックサイクル数のいずれかが最小となるオブジェクトを最適なものとして決定する。

【0047】以上のように構成された本発明に係る翻訳装置の動作について、具体的なソースプログラムが入力された場合を例にとって説明する。図12は、本装置の動作手順を示すフローチャートである。図13は、c言語で書かれたソースプログラム1010のリストである。なお、このソースプログラム1010は、実施形態1における図6に示されたアセンブラープログラムと同一の処理、即ち、メイン関数mからブロックコピーを行う関数fとゼロラン長の比較を行う関数gを順次に呼び出すものである。

【0048】まず、パラメータ獲得部1005は、操作者から分割パラメータ（ここでは、「関数」）と最適パラメータ（ここでは、「コードサイズ」）の指定を受ける（ステップS1201）。ブロック分割部1001は、パラメータ獲得部1005から通知される分割パラメータ（「関数」）に基づいて、図13に示されたソースプログラム1010を3個の関数m, f, gに分割する（ステップS1202）。

【0049】そして、分割された各関数m, f, gごとに、以下の処理が繰り返される（ステップS1203～S1209）。中間コード生成部1002は、各関数m, f, gごとに、中間コードを生成することにより、3個の中間ファイルm, f, gを出力する（ステップS1204）。

【0050】続いて、コード生成部1003は、各中間ファイルm, f, gごとに、命令テーブル1006を参照することにより、8種類のオブジェクトm1～m8、f1～f8、g1～g8を生成する（ステップS1205～S1207）。このとき、コード生成部1003は、命令テーブル1006を参照することによって、命令だけで

なく、各中間コードに対応する全ての命令のサイズ及びクロックサイクル数の合計値をも算出して各オブジェクトに書き込んでおく（ステップS1206）。

【0051】図14は、オブジェクトm1～m8のそれぞれのリストである。8個のオブジェクトm1～m8は全く同一であるので、図14にはそのうちの一つが示されている。これは、中間ファイルmには命令デコーダに依存する中間コードが存在しなかったことによる。なお、このリストの末尾2行は、「このオブジェクトの合計サイズは11であり、合計クロックサイクル数は16である。」ことを示している。

【0052】図15(a)は、オブジェクトf1、即ち、第1命令デコーダ103aをターゲットとする関数fのオブジェクトのリストである。ここでは、ブロック転送の処理がデコーダ非依存型命令“mov”等により実現されていることが分かる。図15(b)は、オブジェクトf8、即ち、第8命令デコーダ103hをターゲットとする関数fのオブジェクトのリストである。ここでは、ブロック転送の処理がデコーダ依存型命令“sp1”により実現されていることが分かる。なお、他の6個のオブジェクトf2～f7のリストは省略されている。

【0053】同様に、図16(a)は、オブジェクトg1、即ち、第1命令デコーダ103aをターゲットとする関数gのオブジェクトのリストである。ここでは、ゼロラン長比較の処理がデコーダ依存型命令“sp1”により実現されていることが分かる。図16(b)は、オブジェクトg8、即ち、第8命令デコーダ103hをターゲットとする関数gのオブジェクトのリストである。ここでは、ゼロラン長比較の処理がデコーダ非依存型命令“mov”等により実現されていることが分かる。

【0054】最後に、オブジェクト決定部1004は、パラメータ獲得部1005から通知される最適パラメータ（サイズ）に基づき、関数m, f, gごとに、コード生成部1003によって生成された8種類のオブジェクトの合計サイズを読み出して比較することにより、その合計サイズが最小となるオブジェクトを1個だけ選択する（ステップS1208）。具体的には、関数mについては、合計サイズに差がないために図14に示されるオブジェクトm1が選択されるが、関数fについては、合計サイズが最小である図15(b)に示されたオブジェクトf8が選択され、関数gについては、合計サイズが最小である図16(a)に示されたオブジェクトg1が選択される。

【0055】このようにして、全ての関数m, f, gのそれぞれについて選択された3個のオブジェクトm1, f8, g1は、オブジェクト決定部1004によって1個に結合された後、図6に示される最終オブジェクトとして出力される（ステップS1210）。以上ようにして、本装置に入力されたソースプログラム1010は、コードサイズを最小にする観点より、関数ごとに、ター

ゲットとすべき命令デコーダが決定され、その決定に基づくオブジェクトが生成された。これにより、従来の單一デコーダ型プロセッサをターゲットとする翻訳装置に比べ、より小さいブロック単位でコードサイズの最適化が行われるため、従来の翻訳装置によって出力されるオブジェクト（例えば、オブジェクトm1+オブジェクトf1+オブジェクトg1）よりもコードサイズの小さいオブジェクトが生成される。

【0056】なお、本実施形態では、分割パラメータが「関数」でかつ最適パラメータが「コードサイズ」と指定されたが、本発明は、このような場合に限定されるものでない。例えば、分割パラメータが「基本ブロック」でかつ最適パラメータが「クロックサイクル数」と指定された場合には、ブロック分割部1001はソースプログラム1010を基本ブロックに分割し、オブジェクト決定部1004は合計サイクル数が最小のオブジェクトを決定するので、これにより、実行時間が最小となるオブジェクトの組合せが最終オブジェクトとして出力されることになる。

【0057】また、最適なオブジェクトを決定する要因として1種類のパラメータに限られることはなく、例えば、命令サイズとクロックサイクル数の合計値を決定要因としてもできる。さらに、本実施形態では、一旦、8種類のオブジェクトを生成した後に（ステップS1206）それらの中から最適なものを選択したが（ステップS1208）、その順序に限られるものではなく、逆の順序であってもよい。

【0058】具体的には、1個の中間ファイルに対して、各中間コードごとに命令テーブルを参照する操作を2回繰り返す。即ち、1回目の参照においては、命令を参照することなく命令サイズだけを参照することによって8種類の命令セットの中から最小サイズとなり得る1個を決定し、続いて、2回目の参照においては、いま決定した命令セットだけを用いて各中間コードに対応する命令だけを参照することによって1個の最終オブジェクトを生成する。これによって、本実施形態の場合と比較し、命令テーブルを参照する操作が1回ではなく2回に増加したものの、8個の暫定的なオブジェクトを生成するステップが回避されることから、翻訳に必要な作業領域（一時的なメモリ領域）の容量が削減される。

【0059】また、本実施形態では、8種類の命令セットは、1個の命令テーブル1006に格納されていたが、8個の命令テーブルに分割されて格納される構成であってもよい。そして、8種類の命令テーブルのそれぞれに、中間コードから命令を生成するための他の必要な情報（例えば、変数の資源への割り付け情報等）を含ませることにより、これらの情報を加味した命令生成が可能となる。

【0060】

【発明の効果】以上の説明から明らかのように、本発明

に係るプロセッサは、命令メモリから命令を読み出して解読する命令解読手段と、その命令解読手段から出力される解読結果に従って演算処理を実行する実行手段とを備えるプロセッサであって、前記命令解読手段は、前記命令メモリから命令を読み出すフェッチ手段と、前記フェッチ手段により読み出された命令を解読し、少なくとも1個の同一命令に対して異なる解読結果を出力するn（2以上の整数）個の命令解読器と、前記n個の命令解読器から選択された一つの命令解読器の解読結果だけが前記実行手段に出力されるよう前記n個の命令解読器を制御する制御手段とからなることを特徴とする。

【0061】これによって、例えば、1バイト長命令であれば実質的に256種類を超える命令を備えるプロセッサ、また、2バイト長命令であれば実質的に65536種類を超える命令を備えるプロセッサが実現されるので、従来の單一デコーダ型プロセッサに比べ、同一処理を行わせる機械語プログラムであってもコードサイズ及び実行時間の短縮化が図られる。

【0062】ここで、前記制御手段は、前記フェッチ手段により読み出された命令が所定の命令解読器切替命令であるかどうかを判断する判断部と、前記判断部により所定の命令解読器切替命令であると判断された場合には、その命令解読器切替命令に基づいて前記選択を行う選択部とからなるとともにできる。これによって、命令解読器は命令によって切り替えられるので、プログラム中に命令解読器切替命令を配置しておくことにより、複数の命令解読器の中から所望の命令解読器を所望のタイミングで有効とさせることができる。

【0063】さらに、前記選択部は、前記判断部により所定の命令解読器切替命令であると判断された場合には、その命令解読器切替命令から一の命令解読器を特定する情報を抽出する抽出部と、前記抽出部により抽出された情報を記憶する記憶部と、前記n個の命令解読器のうち前記記憶部に記憶された情報に対応する命令解読器の解読結果だけが前記実行手段に出力されるよう前記n個の命令解読器を制御する出力制御部とからなるとすることができる。

【0064】そして、前記出力制御部はさらに、前記n個の命令解読器の解読結果が前記実行手段に出力されることを命令解読器ごとに許可又は禁止するゲート部と、前記記憶部に記憶された情報に基づいて前記ゲート部を制御するゲート制御部とからなるとともにできる。これによって、選択された命令解読器は次にいずれかの命令解読器が選択されるまで有効とされ、また、有効とされた1個の命令解読器の解読結果だけがゲート部を通過して出力されるので、現在のプロセッサの状態に応じて複数の命令セットの中から有効とすべき1個の命令セットを決定するという状態遷移を容易に実現することができる。

【0065】ここで、上記命令解読器の数nを2とする

こともできる。これによって、有効な命令解読器を特定するための情報や命令解読器を有効とさせるための制御信号は 1 ビットで足りるので、回路規模の増加を最小限に抑制しつつ上記プロセッサを実現することが可能となる。本発明に係る翻訳装置は、プログラムを翻訳することにより、 n (2 以上の整数) 個の異なる種類の命令解読器を備えるプロセッサを対象とする機械語命令列を生成する翻訳装置であって、最適な機械語命令列を生成するためには前記 n 個の命令解読器の中のいずれを対象とすべきかを決定する決定手段と、前記決定手段により決定された命令解読器を対象として前記プログラムを翻訳し機械語命令列を生成する生成手段とを備えることを特徴とする。

【0066】これによって、複数の命令デコーダ、即ち、複数の命令セットを備えるプロセッサを対象とする翻訳装置が実現され、高級言語等による上記プロセッサの利用が可能となる。また、前記決定手段によって最適な命令解読器が決定された後にその 1 個の命令解読器だけを対象とする機械語命令列が生成されるので、複数の機械語命令列を生成する方法に比べ、翻訳に必要な作業用メモリの容量が削減される。

【0067】ここで、前記決定手段は、生成された機械語命令列のコードサイズが最小となる場合やその機械語命令列を前記プロセッサが実行するのに要する時間が最小となる場合を最適として前記決定を行うことができる。これによって、従来の單一デコーダ型プロセッサを対象とする翻訳装置に比べ、同一処理を行わせるソースプログラムであっても、コードサイズや実行時間がより短縮化された機械語命令列が生成される。

【0068】また、前記翻訳装置はさらに、前記プログラムを所定の基本単位に分割する分割手段を備え、前記決定手段及び前記生成手段は、前記分割手段により分割された基本単位ごとに前記決定及び前記生成を行うことができる。これによって、基本単位よりも小さい処理単位で命令解読器を頻繁に切り替えることから生じる処理速度の低下という不具合の発生が回避される。

【0069】また、本発明に係る翻訳装置は、プログラムを翻訳することにより、 n (2 以上の整数) 個の異なる種類の命令解読器を備えるプロセッサを対象とする機械語命令列を生成する翻訳装置であって、前記プログラムから前記 n 個の命令解読器のそれぞれを対象とする機械語命令列とその最適化に必要な情報を生成する暫定翻訳手段と、前記暫定翻訳手段により生成された前記情報に基づいて前記 n 個の機械語命令列の中から最適な 1 個を決定しその機械語命令列をこの翻訳装置が生成する機械語命令列とする決定手段とを備えるとすることができる。

【0070】これによって、 n 個の機械語命令列と共に各機械語命令列の最適化に必要な情報が生成されるので、例えば、中間コードに対応する機械語命令のテーブ

ル検索と中間コードに対応する最適化情報のテーブル検索という 2 種類のテーブル検索が 1 種類のテーブル検索だけで終了し、翻訳に要する時間が短縮されるという効果がある。

【図面の簡単な説明】

【図 1】本発明の実施の形態 1 における命令解読器切替型プロセッサの構成を示すブロック図である。

【図 2】同プロセッサの切替命令デコーダ 102 の詳細な構成を示すブロック図である。

10 【図 3】同プロセッサが備える切替命令の種類を示す図である。

【図 4】同プロセッサの命令デコーダ 103a ~ 103h の詳細な構成を示すブロック図である。

【図 5】同プロセッサが備える命令セットの一部を示す図である。

【図 6】同プロセッサが実行する対象となるオブジェクトのリストを示す図である。

【図 7】本発明の実施の形態 2 における命令解読器切替型プロセッサの命令解読ブロック 700 の構成を示すブロック図である。

20

【図 8】同プロセッサのデマルチプレクサ 702 の詳細な構成を示すブロック図である。

【図 9】同プロセッサの命令デコーダ 703a ~ 703h の詳細な構成を示すブロック図である。

【図 10】本発明の実施の形態 3 における翻訳装置の構成を示す機能ブロック図である。

【図 11】同翻訳装置の命令テーブル 1006 に保持されている情報の一覧を示す図である。

30

【図 12】同翻訳装置の動作手順を示すフローチャートである。

【図 13】同翻訳装置に入力されるソースプログラム 1010 のリストを示す図である。

【図 14】図 10 に示されたオブジェクト m1 ~ m8 の一つのリストを示す図である。

【図 15】図 15 (a) は、図 10 に示されたオブジェクト f1 のリストを示す図である。図 15 (b) は、図

10 10 に示されたオブジェクト f8 のリストを示す図である。

【図 16】図 16 (a) は、図 10 に示されたオブジェクト g1 のリストを示す図である。図 16 (b) は、図 10 に示されたオブジェクト g8 のリストを示す図である。

【図 17】図 17 (a) は、従来のプロセッサが備える 1 バイト長命令の命令フォーマットを示す図である。図 17 (b) は、従来のプロセッサが備える 2 バイト長命令の命令フォーマットを示す図である。

【符号の説明】

100 命令解読ブロック

101 命令レジスタ

50 102 切替命令デコーダ

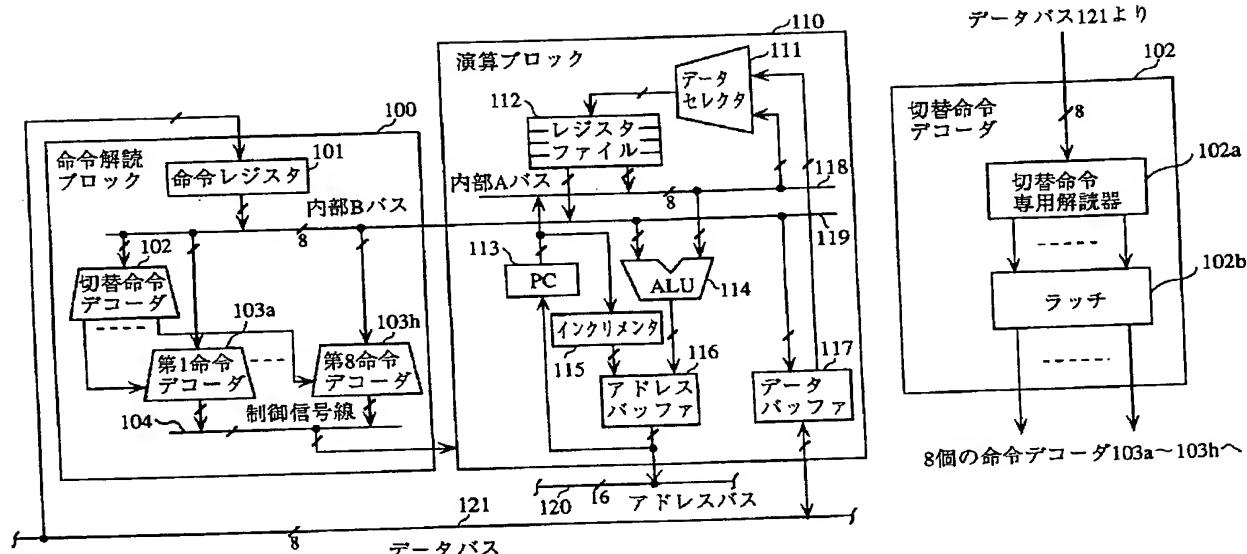
19

- | | |
|-------------------|-----------|
| 1 0 2 a | 切替命令専用解読器 |
| 1 0 2 b | ラッチ |
| 1 0 3 a ~ 1 0 3 h | 命令デコーダ |
| 1 0 4 | 制御信号線 |
| 1 1 0 | 演算ブロック |
| 4 0 1 | プリデコーダ |
| 4 0 2 | シーケンサ |
| 4 0 3 | マイクロROM |

4 0 4	出力バッファ
1 0 0 1	ブロック分割部
1 0 0 2	中間コード生成部
1 0 0 3	コード生成部
1 0 0 4	オブジェクト決定部
1 0 0 5	パラメータ獲得部
1 0 0 6	命令テーブル

【 1】

[図2]

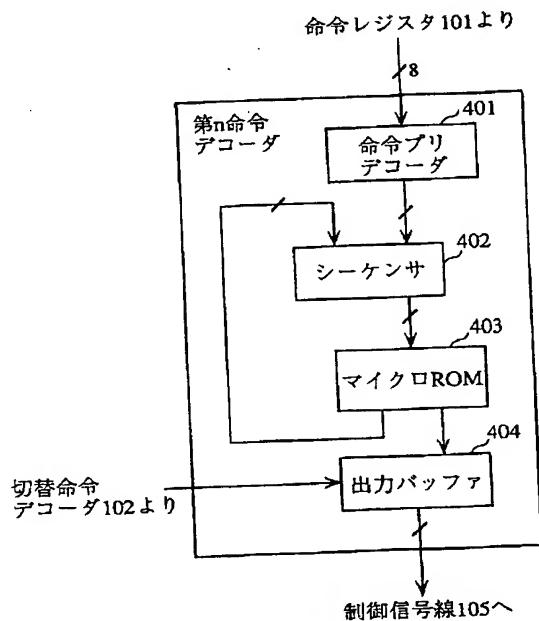
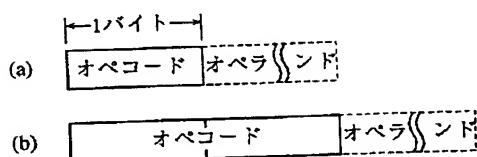


[図3]

【図4】

命令	動作内容
ex1	第1命令デコード103aを有効にさせる
ex2	第2命令デコード103bを有効にさせる
ex3	第3命令デコード103cを有効にさせる
ex4	第4命令デコード103dを有効にさせる
ex5	第5命令デコード103eを有効にさせる
ex6	第6命令デコード103fを有効にさせる
ex7	第7命令デコード103gを有効にさせる
ex8	第8命令デコード103hを有効にさせる

[図 17]



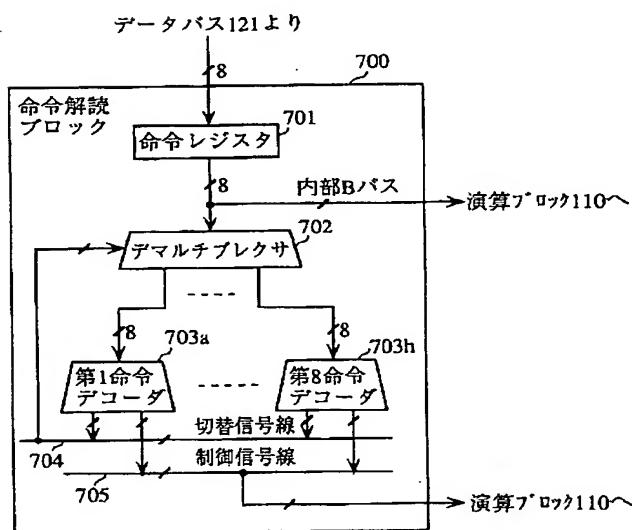
【図 5】

命令	動作内容
add op1,op2	op1をop2に加算
and op1,op2	op1とop2の論理積をとり結果をop2に書き込む
beq op1,op2,label	op1=op2ならばlabelへ分岐
blt op1,op2,label	op1<op2ならばlabelへ分岐
jsr label	labelから始まる関数の呼出し
mov op1,op2	op1をop2に転送
rts	関数からのリターン
shift op1,op2	op1をop2の値だけ右にビットシフト
sp1 op1,op2,op3	[第1命令デコード103aの場合]: op1のゼロラン長をop2に書き込む [第8命令デコード103hの場合]: アドレスop1から始まるメモリの内容をアドレスop2 から始まるメモリにサイズop3分だけブロック転送

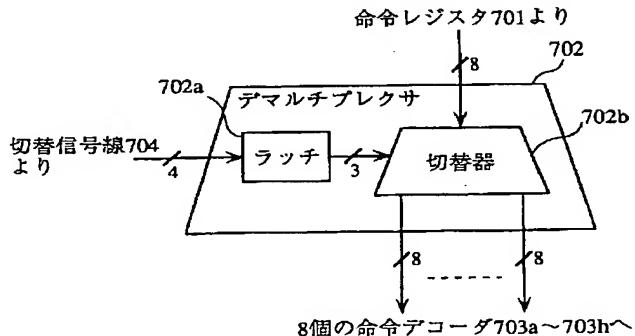
【図 6】

オブジェクト	ステップno.
_m	
mov &a, t1	S600
jsr f	S601
mov &a, t1	S602
jsr g	S603
rst	S604
_f	
ex8	S605
spl &b, t1, 10	S606
rst	S607
_g	
exl	S608
mov 0, i	S609
_L1	
sp1 (&a+i), j	S610
sp1 (&b+i), k	S611
beq k, j, _L3	S612
jsr _error	S613
_L3	
add 1, i	S614
blt i, 10, _L1	S615
rts	S616

【図 7】



【図 8】



【図 14】

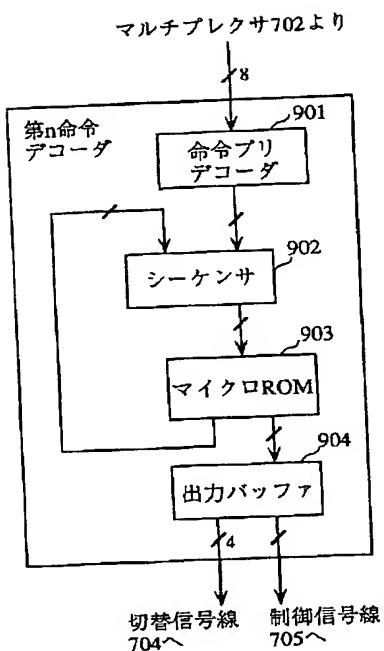
```

_m /* 関数m () */
mov &a, t1 /* 配列aのアドレスをt1に転送 */
jsr f /* _fを呼び出す */
mov &a, t1 /* 配列aのアドレスをt1に転送 */
jsr g /* _gを呼び出す */
rts /* リターン */

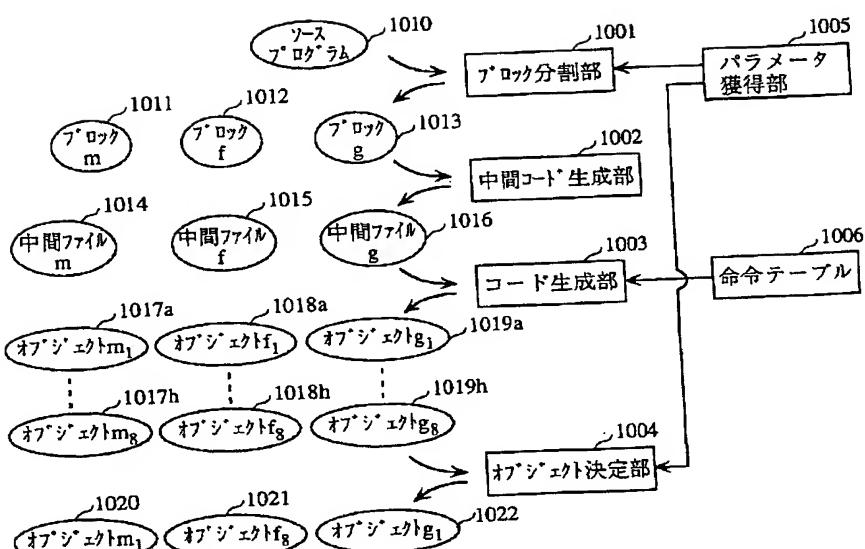
$size = 11 bytes
$cycle = 16

```

【図 9】



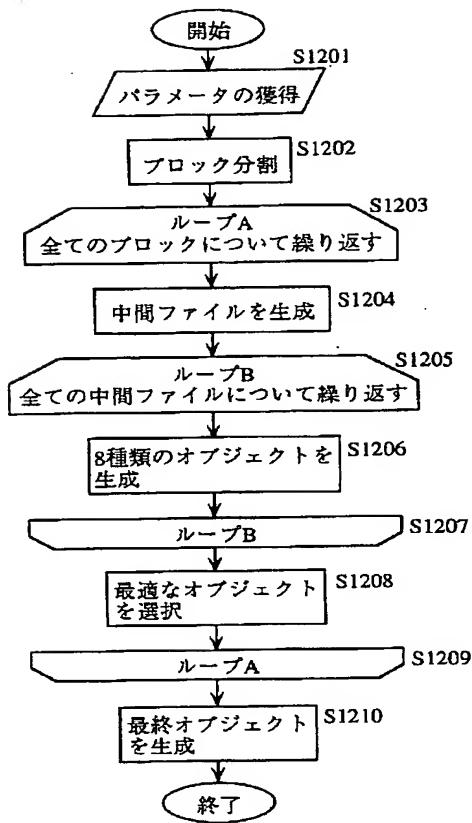
【図 10】



【図 11】

中間コード	命令セット1			命令セット8		
	命令	サイズ	サイクル	命令	サイズ	サイクル
Y=Xadd Y	add x, y	3	5	add x, y	3	5
	≈	≈	≈	≈	≈	≈
	≈	≈	≈	≈	≈	≈
	≈	≈	≈	≈	≈	≈
	≈	≈	≈	≈	≈	≈

【図 1 2】



【図 1 3】

```

char b[10]; /*グローバル配列bを宣言*/
m()
{
    char a[10]; /*ローカル配列aを宣言*/
    f(&a);
    g(&b); /*配列aと配列bの各要素のゼロラン長を比較*/
}

f(char *a)
{
    char i;
    for(i=0; i<10; i++) /*i=0~9まで繰り返す*/
    {
        a[i] = b[i]; /*b[i]をa[i]に代入*/
    }
}

g(char *a)
/*配列aと配列bの各要素の最下位ビットから
   のゼロラン長を比較*/

char i, j, k, t; /*ローカル変数i, j, k, tを宣言*/
for(i=0; i<10; i++) /*i=0~9まで繰り返す*/
{
    t = a[i]; /*a[i]をtに代入*/
    for(j = 0; i & t == 0; t >> 1, j++) /*tの最下位ビットが1にな
      るまで、tを右に1ビットシフトし、そのシ
      フト回数をjとする*/
    t = b[i]; /*b[i]をtに代入*/
    for(k = 0; i & t == 0; t >> 1, k++) /*tの最下位ビットが1にな
      るまで、tを右に1ビットシフトし、そのシ
      フト回数をkとする*/
    if(k != j) error(); /*jとkが一致しなければエラー処理*/
}
}
  
```

This block contains C code for functions f and g. Function f copies elements from array b to array a. Function g compares the zero-run lengths of corresponding elements in arrays a and b by shifting the bits of each element until they differ, counting the number of shifts required for each.

【図 1 5】

(a)

```

-f          /*関数f()*/
exl        /*第1命令デコードを指定*/
mov 0,i    /*ゼロをiに転送*/
_L1
    mov (&b+i),(&a+i) /*配列データb[i]を配列データa
                          [i]に転送*/
    add 1,i    /*iに1を加算*/
    blt i,10,_L1 /*i<10の場合は_L1に分岐*/
    rts        /*リターン*/
$ size = 18 bytes
$ cycle = 24
  
```

(b)

```

-f          /*関数f()*/
ex8        /*第8命令デコードを指定*/
spl &b, t1,10 /*アドレスbからサイズ10のメモリ
                ブロックのデータをアドレスt1から
                サイズ10のメモリブロックに転送*/
    rts        /*リターン*/
$ size = 4 bytes
$ cycle = 6
  
```

【図 16】

(a)

```

-g      ex1      /* 関数g () */
        /* 第1命令デコードを指定 */
        /* ゼロをiに転送 */
_L1    mov 0,i      /* 配列データa[i]の最下位ビットから
        のゼロラン長をjに書き込む */
        /* 配列データb[i]の最下位ビットから
        のゼロラン長をkに書き込む */
        /* j=kの場合は_L2に分岐 */
        /* j=kの場合は_L2に分岐 */
        /* エラー処理 */
_L2    add 1,i      /* iに1を加算 */
        blt i,10,_L1 /* i<10の場合は_L1に分岐 */
        rts          /* リターン */
$ size = 22 bytes
$ cycle = 32

```

(b)

```

-g      ex8      /* 関数g () */
        /* 第8命令デコードを指定 */
        /* ゼロをiに転送 */
_L1    mov 0,i      /* 配列データa[i]をtに転送 */
        mov 0,j      /* ゼロをjに転送 */
_L2    mov t,t2      /* tをt2に転送 */
        and 1,t2     /* t2と1との論理積をt2とする */
        shift 1,t    /* tを右に1ビットシフト */
        add 1,j      /* jに1を加算 */
        beq 0,t2,_L2 /* t2=0の場合は_L2に分岐 */
        mov 0,(&b+i).t /* 配列データb[i]をtに転送 */
        mov 0,k      /* ゼロをkに転送 */
_L3    mov t,t2      /* tをt2に転送 */
        and 1,t2     /* t2と1との論理積をt2とする */
        shift 1,t    /* tを右に1ビットシフト */
        add 1,k      /* kに1を加算 */
        beq 0,t2,_L3 /* t2=0の場合は_L3に分岐 */
        beq k,j,_L4  /* j=kの場合は_L4に分岐 */
        jsr _error   /* エラー処理 */
_L4    add 1,i      /* iに1を加算 */
        blt i,10,_L1 /* i<10の場合は_L1に分岐 */
        rts          /* リターン */
$ size = 46 bytes
$ cycle = 62

```